מבוא לבינה מלאכותית 236501

תרגיל בית מספר 1

מגישים: פבל רסטופצ'ין 311137095 אורי קירשטיין 311137095

פרק ראשון

חלק א

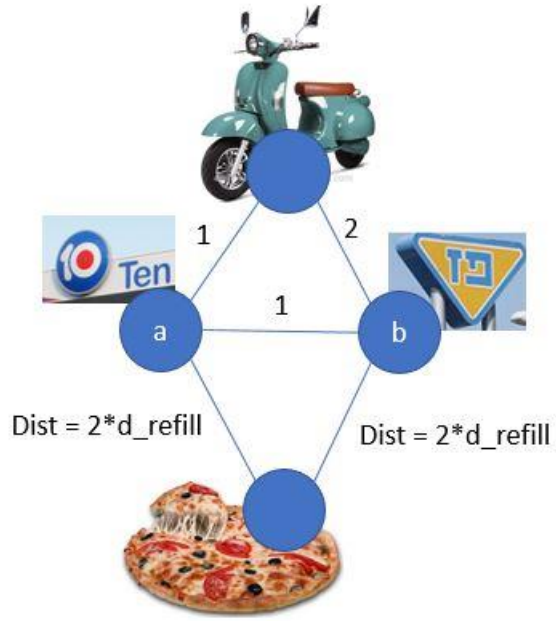
1. להלן הטבלה המבוקשת:

K	K!	K!*L^(K-1)
1	1	1
2	2	10
3	6	150
4	24	3000
5	120	75000
6	720	2250000
7	5040	78750000
8	40320	3150000000
9	362880	1.4175E+11
10	3628800	7.0875E+12

חלק ג

2. מקדם הסיעוף המינימלי האפשרי במרחב החיפוש הוא 0 – יתקבל למשל כאשר נתחיל בלי דלק כלל. לא נוכל להגיע לאף מצב אחר במרחב והגרף יישאר בעל צומת בודד. מקדם הסיעוף המקסימלי האפשרי במרחב החיפוש הוא $|k+l|$, שיתקבל, למשל, אם נוכל להגיע לכל המצבים מנקודת ההתחלה. נוכיח כי זהו המקדם המקסימלי בשלילה. נניח כי קיים מקדם סיעוף $w > |l+k|$. כלומר, קיים מצב במרחב ממנו ניתן להגיע ל w מצבים אחרים. אבל יש רק $|k+l|$ צמתים אחרים לכל היותר בגרף אליהם ניתן להגיע בעזרת האופרטור. סתירה.

3. ייתכנו מעגלים במרחב המצבים שלנו. למשל, עבור המרחב המתואר למטה, נסתכל על סדרת המצבים.



מצב התחלתי - $S_0 = \{v_0, d_{full}, T = \{1\}, F = \emptyset\}$

אם נמשיך לתחנת הדלק הקרובה ביותר נגיע למצב $S_1 = \{v_a, d_{full}, T = \{1\}, F = \emptyset\}$

ממנו נמשיך לתחנת הדלק השנייה $S_2 = \{v_b, d_{full}, T = \{1\}, F = \emptyset\}$.

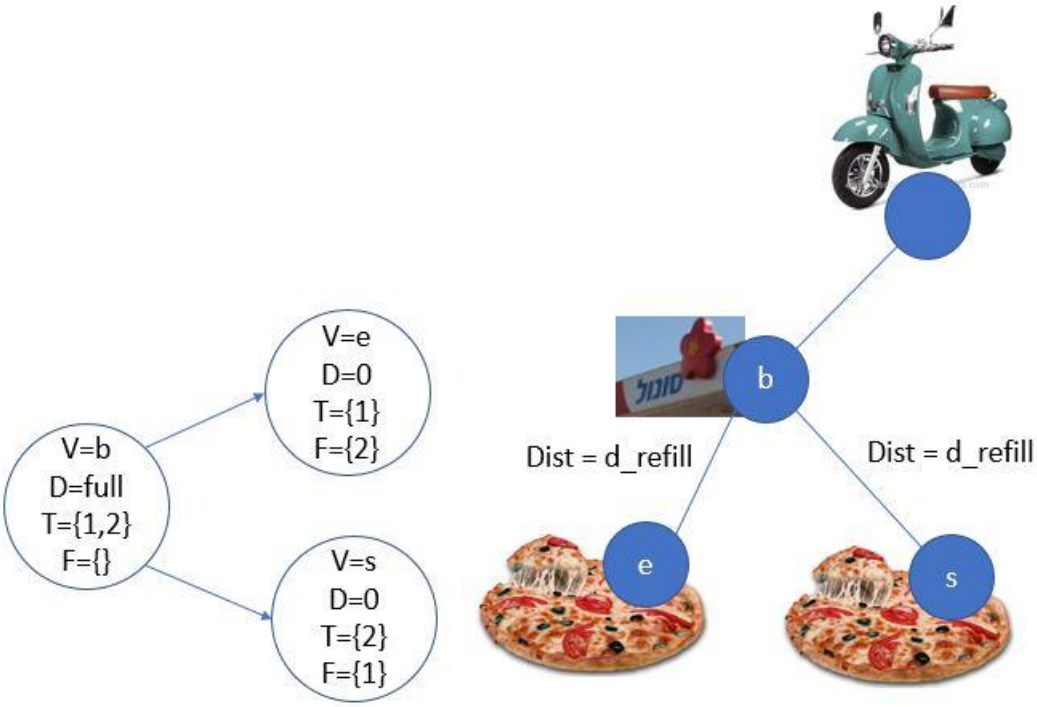
עכשיו יש רק אופרטור אפשרי אחד – ניתן לחזור לתחנת הדלק הקודמת בלבד, ונגיע שוב למצב הקודם,

$$S_1 = \{v_1, d_{full}, T = \{1\}, F = \emptyset\}$$

הגענו לאותו הצומת פעמיים, לכן מצאנו מעגל כנדרש.

4. ישנם אינסוף מצבים במרחב. אמנם מספר האפשרויות השונות לT חסום ע"י $2^{|K|}$, F נקבע ע"י בחירת T באופן יחיד, וכן מספר הצמתים בגרף הוא $|l + k + 1|$, מספר האפשרויות לבחירת d אינסופי כי d פרמטר רציף. לא כל המצבים ישיגים, למשל במפה מהסעיף הקודם שום מצב בצומת התחתון אינו ישיג.

5. ייתכנו בורות שאינם מצבי מטרה במרחב המצבים. למשל במפה שלמטה – המצבים בצמתים e ו s הם בורות, כיוון שבכל דרך בה נגיע אליהם מיכל הדלק יהיה ריק ולא נוכל להמשיך לאף צומת אחר, לכן לא יהיו יותר מעברים חוקיים.



6. $$Succ\big((v_1, d_1, T_1, F_1)\big) = \left\{ (v_2, d_2, T_2, F_2) \,\middle|\, \left\{ \begin{matrix} d_2 = d_{full} \\ v_2 \in l \\ T_2 = T_1 \\ F_2 = F_1 \end{matrix} \right\} \right\} \cup \left\{ (v_2, d_2, T_2, F_2) \,\middle|\, \left\{ \begin{matrix} d_2 = d_1 - dist(v_1, v_2) \geq 0 \\ v_2 \in k \\ T_2 = T_1 \setminus \{v_2\} \\ F_2 = F_1 \cup \{v_2\} \end{matrix} \right\} \right\}$$

7. חסם תחתון מינימלי הוא מספר ההזמנות. במקרה הכי טוב, בכל צעד נגיע לצומת הזמנה אחד, ולא ניאלץ לתדלק כלל. לכן החסם הוא $|k|$

חלק ד

load_map_from_csv: 3.34sec

Solve the map problem.

Map(src: 54 dst: 549) UniformCost time: 1.51 #dev: 17355 total_cost: 7465.52897 |path|: 137

path: [54, 55, 56, 57, 58, 59, 60, 28893, 14580, 14590, 14591, 14592, 14593, 81892, 25814, 81, 26236, 26234, 1188, 33068, 33069, 33070, 15474, 33071, 5020, 21699, 33072, 33073, 33074, 16203, 9847, 9848, 9849, 9850, 9851, 335, 9852, 82906, 82907, 82908, 82909, 95454, 96539, 72369, 94627, 38553, 72367, 29007, 94632, 96540, 9269, 82890, 29049, 29026, 82682, 71897, 83380, 96541, 82904, 96542, 96543, 96544, 96545, 96546, 96547, 82911, 82928, 24841, 24842, 24843, 5215, 24844, 9274, 24845, 24846, 24847, 24848, 24849, 24850, 24851, 24852, 24853, 24854, 24855, 24856, 24857, 24858, 24859, 24860, 24861, 24862, 24863,

24864, 24865, 24866, 82208, 82209, 82210, 21518, 21431, 21432, 21433, 21434, 21435, 21436, 21437, 21438, 21439, 21440, 21441, 21442, 21443, 21444, 21445, 21446, 21447, 21448, 21449, 21450, 21451, 621, 21452, 21453, 21454, 21495, 21496, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549]

חלק ה

.11

Solve the map problem.

Map(src: 54 dst: 549) UniformCost time: 1.14 #dev: 17355 total_cost: 7465.52897 |path|: 137

path: [54, 55, 56, 57, 58, 59, 60, 28893, 14580, 14590, 14591, 14592, 14593, 81892, 25814, 81, 26236, 26234, 1188, 33068, 33069, 33070, 15474, 33071, 5020, 21699, 33072, 33073, 33074, 16203, 9847, 9848, 9849, 9850, 9851, 335, 9852, 82906, 82907, 82908, 82909, 95454, 96539, 72369, 94627, 38553, 72367, 29007, 94632, 96540, 9269, 82890, 29049, 29026, 82682, 71897, 83380, 96541, 82904, 96542, 96543, 96544, 96545, 96546, 96547, 82911, 82928, 24841, 24842, 24843, 5215, 24844, 9274, 24845, 24846, 24847, 24848, 24849, 24850, 24851, 24852, 24853, 24854, 24855, 24856, 24857, 24858, 24859, 24860, 24861, 24862, 24863, 24864, 24865, 24866, 82208, 82209, 82210, 21518, 21431, 21432, 21433, 21434, 21435, 21436, 21437, 21438, 21439, 21440, 21441, 21442, 21443, 21444, 21445, 21446, 21447, 21448, 21449, 21450, 21451, 621, 21452, 21453, 21454, 21495, 21496, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549]

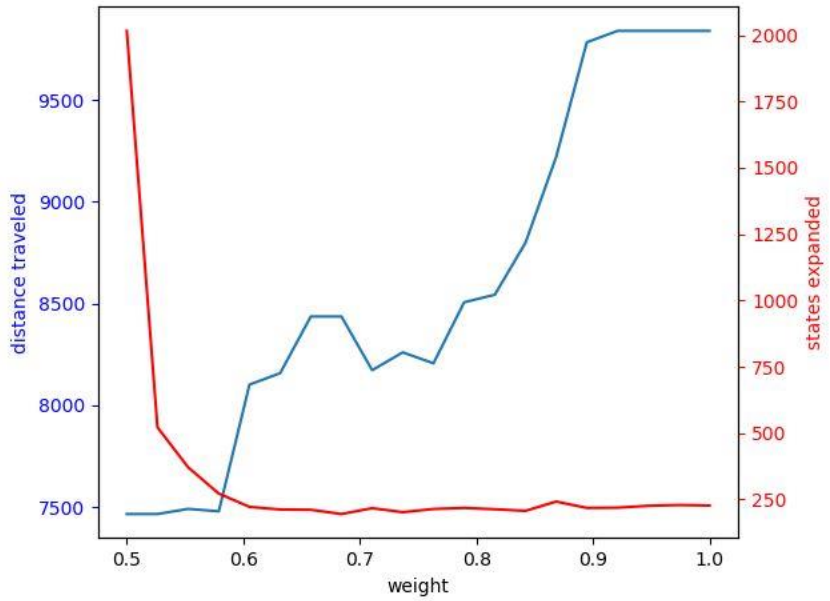
Map(src: 54 dst: 549) A* (h=0, w=0.500) time: 1.16 #dev: 17355 total_cost: 7465.52897 |path|: 137

path: [54, 55, 56, 57, 58, 59, 60, 28893, 14580, 14590, 14591, 14592, 14593, 81892, 25814, 81, 26236, 26234, 1188, 33068, 33069, 33070, 15474, 33071, 5020, 21699, 33072, 33073, 33074, 16203, 9847, 9848, 9849, 9850, 9851, 335, 9852, 82906, 82907, 82908, 82909, 95454, 96539, 72369, 94627, 38553, 72367, 29007, 94632, 96540, 9269, 82890, 29049, 29026, 82682, 71897, 83380, 96541, 82904, 96542, 96543, 96544, 96545, 96546, 96547, 82911, 82928, 24841, 24842, 24843, 5215, 24844, 9274, 24845, 24846, 24847, 24848, 24849, 24850, 24851, 24852, 24853, 24854, 24855, 24856, 24857, 24858, 24859, 24860, 24861, 24862, 24863, 24864, 24865, 24866, 82208, 82209, 82210, 21518, 21431, 21432, 21433, 21434, 21435, 21436, 21437, 21438, 21439, 21440, 21441, 21442, 21443, 21444, 21445, 21446, 21447, 21448, 21449, 21450, 21451, 621, 21452, 21453, 21454, 21495, 21496, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549]

Map(src: 54 dst: 549) A* (h=AirDist, w=0.500) time: 0.34 #dev: 2016 total_cost: 7465.52897 |path|: 137

path: [54, 55, 56, 57, 58, 59, 60, 28893, 14580, 14590, 14591, 14592, 14593, 81892, 25814, 81, 26236, 26234, 1188, 33068, 33069, 33070, 15474, 33071, 5020, 21699, 33072, 33073, 33074, 16203, 9847, 9848, 9849, 9850, 9851, 335, 9852, 82906, 82907, 82908, 82909, 95454, 96539, 72369, 94627, 38553, 72367, 29007, 94632, 96540, 9269, 82890, 29049, 29026, 82682, 71897, 83380, 96541, 82904, 96542, 96543, 96544, 96545, 96546, 96547, 82911, 82928, 24841, 24842, 24843, 5215, 24844, 9274, 24845, 24846, 24847, 24848, 24849, 24850, 24851, 24852, 24853, 24854, 24855, 24856, 24857, 24858, 24859, 24860, 24861, 24862, 24863, 24864, 24865, 24866, 82208, 82209, 82210, 21518, 21431, 21432, 21433, 21434, 21435, 21436, 21437, 21438, 21439, 21440, 21441, 21442, 21443, 21444, 21445, 21446, 21447, 21448, 21449, 21450, 21451, 621, 21452, 21453, 21454, 21495, 21496, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549]

.12



כלל שהמשקל מתקרב ל-1, כך האלגוריתם חמדן יותר, כלומר הוא מסתמך יותר על היוריסטיקה. ככל שההסתמכות על היוריסטיקה גדולה יותר, הפתרון יתקבל מהר יותר אך יהיה פחות אופטימלי.

חלק ו

14. היוריסטיקה זו קבילה. כדי להגיע למצב מקבל נצטרך לבקר בכל נקודות ההזמנה. בפרט, ניאלץ לבקר בנקודת ההזמנה הרחוקה ביותר מאיתנו. המרחק לנקודת הזמנה זו הוא לכל הפחות המרחק האווירי שלנו ממנה.

.16

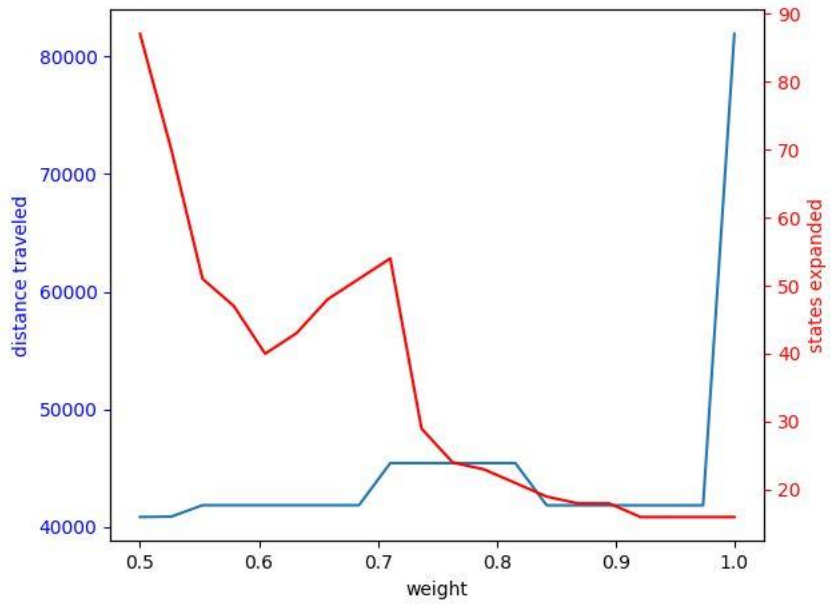
Solve the relaxed deliveries problem.

RelaxedDeliveries(big_delivery) A* (h=MaxAirDist, w=0.500) time: 9.28 #dev: 3908 total_cost: 40844.21165 |path|: 11 path: [33919, 18409, 77726, 26690, 31221, 63050, 84034, 60664, 70557, 94941, 31008] gas-stations: [31221, 70557]

.17

RelaxedDeliveries(big_delivery) A* (h=MSTAirDist, w=0.500) time: 2.48 #dev: 87 total_cost: 40844.21165 |path|: 11 path: [33919, 18409, 77726, 26690, 31221, 63050, 84034, 60664, 70557, 94941, 31008] gas-stations: [31221, 70557]

.18

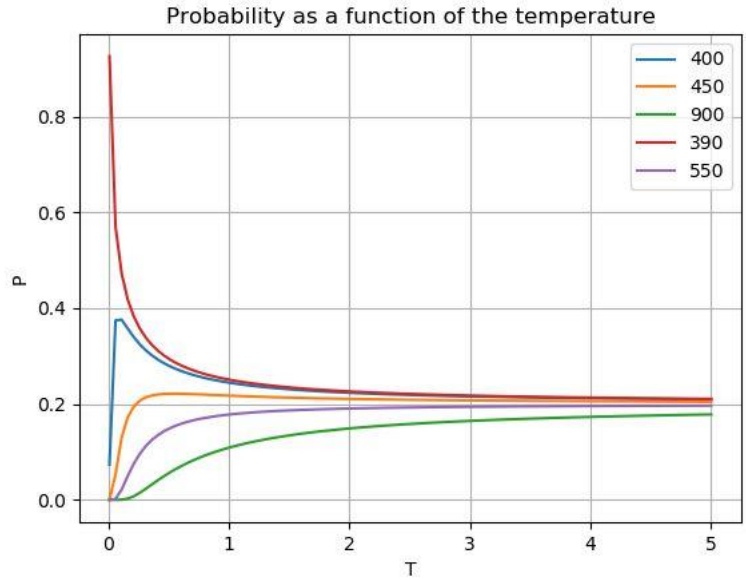


חלק ז

19. יהי $x^t = \left[h_1, h_2, \dots h_{\min(N,|OPEN|)}\right]$. לפי ההגדרה, $\forall x_i \in x^t: \Pr(x_i) = \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points}\left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}}$. נבצע שינוי סקלה:
נכפול את כל איברי הוקטור בקבוע k שאינו 0: $X^t = \left[kh_1, kh_2, \dots kh_{\min(N,|OPEN|)}\right]$. נציב בהגדרה ונקבל:

$$\forall x_i \in X^t: \Pr(x_i) = \frac{\left(\frac{kx_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points}\left(\frac{kx_h}{\alpha}\right)^{-\frac{1}{T}}} = \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}(k)^{-\frac{1}{T}}}{(k)^{-\frac{1}{T}}\sum_{pnt_h \in best\ N\ points}\left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}} = \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points}\left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}}.$$

.20

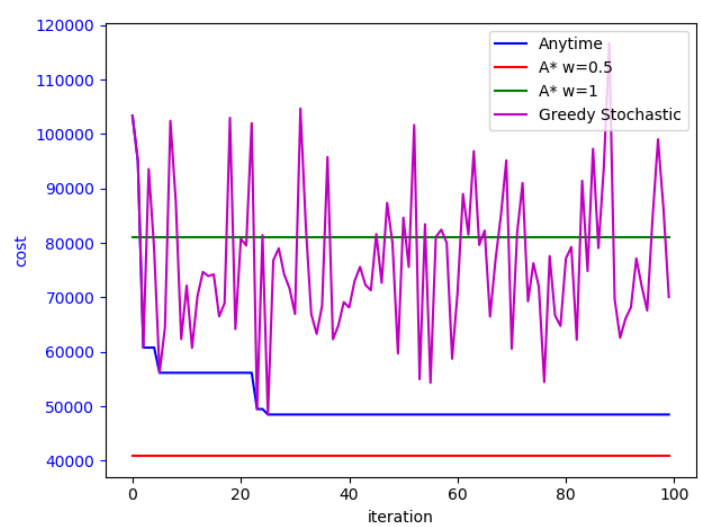


.21 מהתבוננות בגרף, $\lim_{T\to 0} \Pr(x_i) = \lim_{T\to 0} \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} \left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}} = \begin{cases} 1\ when\ x_i = \alpha \\ 0\ otherwise \end{cases}$

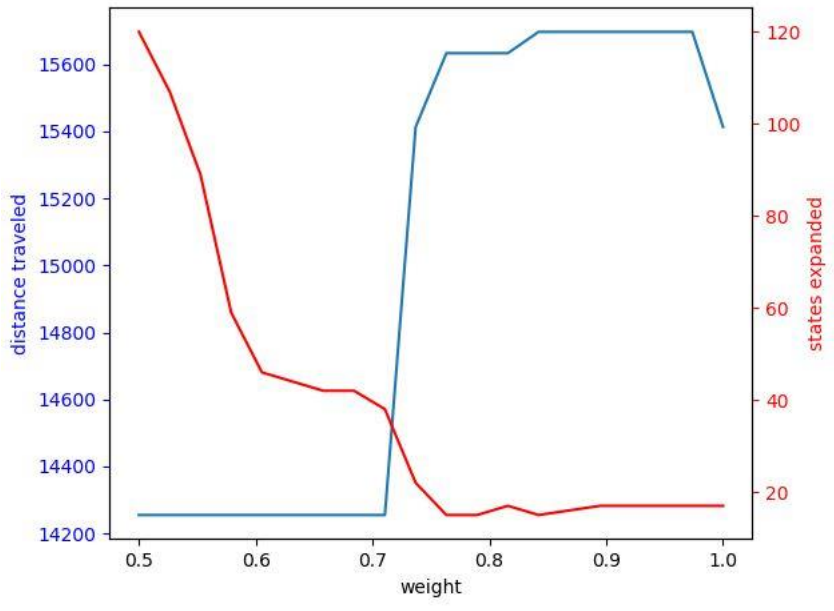
.22

$$\lim_{T\to\infty} \Pr(x_i) = \lim_{T\to\infty} \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} \left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}} = \frac{\left(\frac{x_i}{\alpha}\right)^{0}}{\sum_{pnt_h \in best\ N\ points} \left(\frac{x_h}{\alpha}\right)^{0}} = \frac{1}{\sum_{pnt_h \in best\ N\ points} 1}$$
$$= \frac{1}{\min(N, |OPEN|)}$$

.24 להלן הגרף המתקבל:



.26



27. נשתמש בהיוריסטיקה – מרחק בבעיית ה RelaxedDeliveriesProblemState. הנתונים לבעיה הם:

צמתי תחנות דלק – זהים לבעיה האמיתית

תחנות עצירה שיש לבקר בהן – תחנות עצירה שעוד לא ביקרנו בהן בבעיה האמיתית

גודל מיכל דלק מלא – זהה לבעיה האמיתית

צומת התחלה – הצומת שעליו אנו בודקים את ערך ההיוריסטיקה

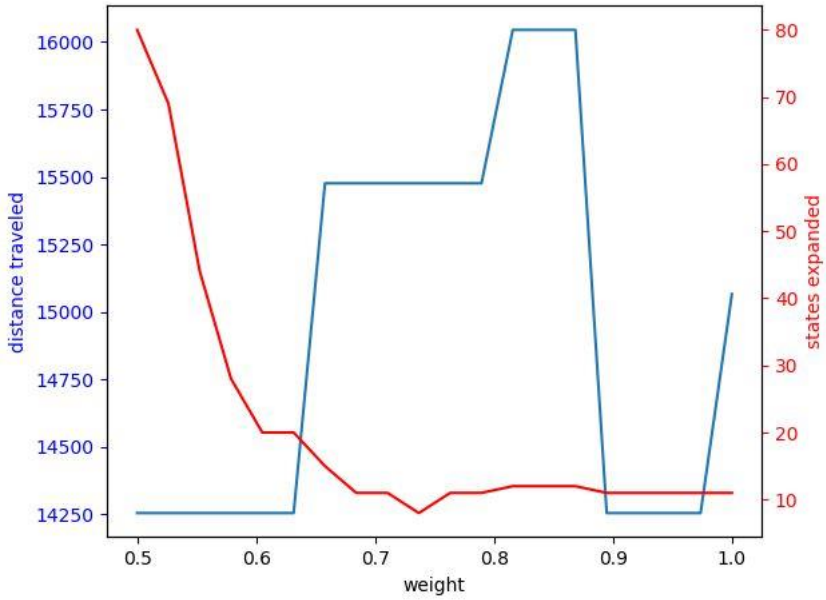
דלק התחלתי – זהה לדלק שיש ברשותנו בבעיה האמיתית

אם נמצא פתרון נחזיר את המחיר שלו (ואנחנו יודעים שהוא יהיה הפתרון האופטימלי לבעיה). אם לא נמצא, נחזיר אינסוף – הבעיה אינה פתירה.

היוריסטיקה זו קבילה כיוון שהיא מניחה מרחק אווירי בין כל זוג צמתים בגרף. המרחק בפועל בין כל 2 צמתים הוא תמיד לכל הפחות המרחק האווירי ביניהן.

כדי למצוא מרחק זה נשתמש באלגוריתם *A עם היוריסטיקת MSTAirDist כאשר המשקל הוא 0.5. כפי שהוכח בכיתה, מובטח שהאלגוריתם ייתן פתרון אופטימלי.

.28



בהשוואה לסעיף 26, עבור משקל 0.5, רואים כי ההיוריסטיקה החדשה פיתחה 80 מצבים בלבד לעומת 120 ולכן היא ההיוריסטיקה העדיפה. בכל משקל שהוא, כמו המצבים שפותחו בסעיף 26 הייתה גדולה יותר מאשר הגרף בסעיף הנוכחי. עם זאת רואים כי האורך המקסימלי המוחזר בסעיף 26 קטן מ15,800 לעומת 16,000 בגרף של סעיף זה.

בסעיף 26 מספר הפיתוחים ירד לראשונה מתחת ל80 עבור משקל 0.579 והיה 59. הפתרון במשקל זה לא פוגע באיכות הפתרון. ניתן לראות זאת ישירות מהגרף של סעיף 26, שם נקודה זו נופלת על החלק הישר של עלות המסלול.

פרק שני

סעיף א –

נתונה פונקציה היוריסטית חלקית h(s), כך ש$h^*(s) \geq h(S) \geq 0$. נסתכל על $h_0(s)$:

- מקרה ראשון: $Applicable_h(s) = True$. כאן $h_0(s) = h(s) \leq h^*(s)$.
- מקרה שני: $Applicable_h(s) = False$. כאן $0 = h_0(s) \leq h(s) \leq h^*(s)$.

סעיף ב –

```
def h1(State s):
    if ApplicableH(s) is True:
        return h0(s)
    if is_goal(s):
        return 0
    else
        ret = -1
        for succsessor in s.succsessors_list:
            if ApplicableH(succsessor) is False:
                return 0
            else:
                succsessor_cost = h(successor)
                if succsessor_cost < ret or ret == -1:
                    ret = succsessor_cost
        if ret == -1:
            return infinity
        return ret
```

נוכיח כי ההיוריסטיקה קבילה ומיודעת יותר מהיוריסטיקה h_0.

קבילות –

- אם הצומת הוא צומת מטרה, מתקיים $h^*(s) = 0$, ולכן $h_1(s) = 0 \leq 0 = h^*(s)$
- כאשר $Applicable_h(s) = True$, אז $h_1(s) = h(s) \leq h^*(s)$
- כאשר $Applicable_h(s) = False$ והצומת הוא בור, אין דרך להגיע ממנו לצומת מטרה. לכן, $h^*(s) = \infty$ ומתקיים $h_1(s) = \infty \leq h^*(s)$ (כמובן ההנחה היא שיש ייצוג יחיד לאינסוף).
- כאשר $Applicable_h(s) = False$, ולפחות לאחד הבנים של הצומת מתקיים $Applicable_h(s') = False$, $h_1(s) = 0 \leq h^*(s)$
- כאשר $Applicable_h(s) = False$, ולכל הבנים מתקיים $Applicable_h(s') = True$:
  נסתכל על $h^*(s)$. כיוון שהצומת איננו צומת מטרה, המסלול שעלותו מינימלית עובר דרך אחד הבנים, נסמנו t.
  מתקיים כי:
  $$h^*(s) = h^*(t) + cost(s,t) \geq h^*(t) \geq \min_{u \in sucessors(s)} h^*(u) \geq \min_{u \in sucessors(s)} h(u) = h_1(s)$$

מיודעות –

- כאשר $Applicable_h(s) = False$, תמיד הערך שהיוריסטיקה h1 מחזירה אי-שלילי, ולכן מיודעת יותר מh0 שתחזיר רק ערך 0 במקרה זה.
- אחרת, שתי ההיוריסטיקות יחזירו ערכים זהים, $h(s)$.

סעיף ג –

```
def h2(State s):
    if ApplicableH(s) is True:
        return h0(s)
    if is_goal(s):
        return 0
    else
        ret = -1
        for succsessor in s.succsessors_list:
            if successor in s.closed or successor == s:
                continue
            if ApplicableH(succsessor) is False:
                return 0
            else:
                succsessor_cost = h(successor)
                if succsessor_cost < ret or ret == -1:
                    ret = succsessor_cost
        if ret == -1:
            return infinity
        return ret
```

האלגוריתם זהה לאלגוריתם בסעיף הקודם למעט המקרה שיש קשת אחת או יותר לצומת בו ביקרנו או לולאה עצמית אחת או יותר בצומת. כל הוכחות לכל שאר המקרים עדיין תקפות כמו במקרה הקודם.

טענה: המסלול האופטימלי לעולם אינו עובר דרך מעגל (לרבות לולאה עצמית). הוכחה: ידוע כי עלות כל קשת היא לפחות δ. נניח בשלילה כי קיים מסלול העובר דרך לולאה עצמית. נלך באותו המסלול בדיוק, רק שלא נעבור בלולאה העצמית. המסלול החדש קצר יותר לפחות בקשת אחת, לכן המחיר זול יותר לפחות ב δ, בסתירה למינימליות.

לכן, ניתן להתעלם מקשתות עצמיות ומקשתות אחוריות לחלוטין בחישוב ההיוריסטיקה. לשים לב שמקרה של צומת שיוצאות ממנו רק לולאות אחוריות או עצמיות, האלגוריתם יזהה אותו כבור. נשים לב שכאן ניאלץ לתחזק מאגר של צמתים שכבר ביקרנו בהן, כמו במימוש הסטנדרטי של אלגוריתם *A. במקרה הקודם לא היינו זקוקים לכך, כי אין מעגלים בעץ.

סעיף ד –

קיים אלגוריתם כזה – *IDA עם ההיורסטיקה $h_0(h', s)$, כאשר החסם הראשון הוא הערך ההיוריסטי של מצב ההתחלה. מספר הצעדים חסום ע"י $\frac{h'(s_{inital})}{\delta}$, לכן מובטח כי האלגוריתם ימצא את הפתרון האופטימלי. ברגע ש*IDA ימצא מסלול למצב מטרה, זה יהיה מסלול אופטימלי וניתן להפסיק לחפש.

כיוון שמרחב המצבים הוא עץ מכוון, אין בו מעגלים. אלגוריתם *A ואלגוריתם *IDA יפתחו את המסלול באופן דומה.

באלגוריתם *A ייתכן כי האלגוריתם ימשיך לחפש גם אחרי שנמצא מסלול למצב מטרה, אם בתור המצבים OPEN מצב המטרה לא יהיה הראשון.